Stochastic Machine Scheduling with Precedence Constraints*

Martin Skutella
Fakultät II, Institut für Mathematik, Sekr. MA 6-1, Technische Universität Berlin, 10623 Berlin, Germany
skutella@math.tu-berlin.de

Marc Uetz
Faculty of Economics and Business Administration, Quantitative Economics, Universiteit Maastricht, 6200 MD Maastricht, The Netherlands, m.uetz@ke.unimaas.nl

Abstract

We consider parallel, identical machine scheduling problems where the jobs are subject to precedence constraints, release dates, and the processing times of jobs are governed by independent probability distributions. The objective is to minimize the expected value of the total weighted completion time $\sum w_j C_j$, where $w_j \geqslant 0$. Building upon a linear programming relaxation by Möhring, Schulz, and Uetz (1999) and an idle time charging scheme by Chekuri, Motwani, Natarajan, and Stein (2001), we derive the first constant-factor approximation algorithms for this model.

1 Introduction

This paper addresses stochastic parallel machine scheduling problems with the objective to minimize the total weighted completion time in expectation. Machine scheduling problems have attracted researchers for decades since they play an important role in various applications from the areas of operations research, management science, and computer science. The total weighted completion time objective is of particular importance in scheduling environments where many jobs are to be scheduled on a limited number of machines, and a good average performance is desired. Prominent examples for such a scheduling situation are problems that arise, e.g., in compiler optimization (Chekuri et al. 1996) and in parallel computing (Chakrabarti and Muthukrishnan 1996). The main characteristic of *stochastic* scheduling problems is the fact that part of the input data, in this paper the processing times of the jobs, may be subject to random fluctuations. Hence, the effective processing times are not known with certainty in advance. This assumption is of particular practical relevance in many applications.

Denote by $V = \{1,\dots,n\}$ a set of jobs which must be scheduled on m parallel, identical machines. Each machine can handle only one job at a time, and the jobs can be scheduled on any of the machines. Once the processing of a job is started on one machine, it must be processed without preemption on this machine. Precedence constraints are given by an acyclic digraph $G = (V,A)$, where any arc $(i,j) \in A$ restricts the start time of job j to be not earlier than the completion time of job i. We consider problems with and without release dates r_j for the jobs, with the intended meaning that job j must not be started earlier than r_j. In the classical (deterministic) setting, the objective is to minimize the total weighted completion time $\sum_{j\in V} w_j C_j$, where w_j is a non-negative weight and C_j denotes the completion time of job j.

In the stochastic model, it is assumed that the processing time p_j of a job j is not known in advance. It becomes known only upon completion of the job. However, the distribution of the corresponding random variable P_j is given beforehand. Let $P = (P_1,\dots,P_n)$ denote the vector of random variables for the processing times, and denote by $p = (p_1,\dots,p_n)$ a particular realization of the processing times. By $\mathrm{E}[P_j]$ we denote the expected processing time

*An extended abstract of this work appeared in: Proceedings of the Twelfth Annual ACM–SIAM Symposium on Discrete Algorithms (SODA 2001), Washington DC, 2001, pp. 589–590. This work was supported in part by the EU Thematic Networks APPOL I+II, Approximation and Online Algorithms, IST-1999-14084 and IST-2001-30012, and by the DFG Research Center 'Mathematics for key technologies: Modelling, simulation and optimization of real-world processes'. The second author was supported by the German-Israeli Foundation for Scientific Research and Development (GIF) under grant I 246-304.02/97, and by the Deutsche Forschungsgemeinschaft (DFG) under grant Mo 446/3-4.

of a job j. We assume throughout that the processing times of the jobs are stochastically independent. In the classical three-field notation of Graham, Lawler, Lenstra, and Rinnooy Kan (1979), the problem of minimizing the expected total weighted completion time can be denoted by $\mathrm{P}|prec, r_j|\mathrm{E}\left[\sum w_j C_j\right]$.

In fact, the twist from deterministic to stochastic processing times changes the nature of the scheduling problem considerably. The solution of a stochastic scheduling problem is no longer a simple schedule, but a *scheduling policy*. We adopt the notion of scheduling policies as proposed by Möhring et al. (1984). Roughly spoken, a scheduling policy makes scheduling decisions at certain decision times t, and these decisions are based upon the observed past up to time t as well as the a priori knowledge of the input data of the problem. The policy, however, must not anticipate information about the future, such as the actual realizations p_j of the processing times of the jobs which have not yet been completed by time t.

In general, scheduling policies can be rather complex in the sense that they exploit the stochastic information that evolves over time. The simplest, yet practically attractive scheduling policies are *list scheduling policies*. Graham's classical list scheduling algorithm (Graham 1966) is perhaps the most natural among them: Given a priority list L of the jobs, at any time it greedily schedules the first available job(s) from the list. A job is *available* as soon as it is released and all of its predecessors are completed. If precedence constraints or release dates exist, it may thus happen that the order of start times of jobs differs from the order of the jobs in the given priority list; the jobs are scheduled 'out of order' with respect to the given priority list L. For the deterministic problem $\mathrm{P}|prec|C_{\max}$ with *makespan* objective, it is well known that Graham's list scheduling algorithm achieves a performance guarantee of $2-1/m$ for any priority list of the jobs (Graham 1966). This result can be established for stochastic processing times as well (Chandy and Reynolds 1975). However, even in the deterministic setting there are examples which show that the performance of Graham's algorithm can be arbitrarily bad for the total weighted completion time objective $\sum w_j C_j$ (Schulz 1996). For the deterministic problem $\mathrm{P}|r_j, prec|\sum w_j C_j$, the currently best known performance guarantee of 4 relies on another, so-called *job-based* list scheduling algorithm (Munier, Queyranne, and Schulz 1998). For stochastic processing times, approximation results for parallel machines were previously known for problems without precedence constraints only (Möhring et al. 1999).

The results derived in this paper rely on a list scheduling algorithm which generalizes both Graham's and job-based list scheduling. It has been suggested by Chekuri et al. (2001) to obtain a 5.828-approximation for the deterministic problem $\mathrm{P}|r_j, prec|\sum w_j C_j$. The basic idea is to extend Graham's list scheduling in such a way that a job may be scheduled out of order only if 'enough' deliberate idle time has accumulated before. We show that an appropriate adaption of the list scheduling algorithm of Chekuri et al. (2001), based upon a priority list that is derived from an optimal solution to a generalized LP-relaxation by Möhring et al. (1999), leads to performance guarantees also for the stochastic model.

The table below summarizes known performance guarantees for stochastic parallel machine scheduling problems with the total weighted completion time objective. While the results in the first four rows are from Möhring et al. (1999), the remaining results are derived in this paper. In the table, $\sqrt{\Delta}$ is an upper bound on the coefficient of variation $\mathrm{CV}[P_j]$ of the processing time distributions P_j, the number of machines is denoted by m, and β is an arbitrary non-negative parameter. The last column shows the respective performance bounds for processing time distributions where the coefficient of variation is bounded by 1, such as exponential, uniform, or Erlang distributions.

Table 1: Performance bounds for stochastic machine scheduling problems.

scheduling model	performance guarantee	
	arbitrary P_j	$\mathrm{CV}[P_j] \leqslant 1$
$1\|prec\|\mathrm{E}\left[\sum w_j C_j\right]$	2	2
$1\|r_j, prec\|\mathrm{E}\left[\sum w_j C_j\right]$	3	3
$\mathrm{P}\|\|\mathrm{E}\left[\sum w_j C_j\right]$	$1+\frac{(m-1)(\Delta+1)}{2m}$	$2-\frac{1}{m}$
$\mathrm{P}\|r_j\|\mathrm{E}\left[\sum w_j C_j\right]$	$3-\frac{1}{m}+\max\{1, \frac{m-1}{m}\Delta\}$	$4-\frac{1}{m}$
$\mathrm{P}\|in\text{-}forest\|\mathrm{E}\left[\sum w_j C_j\right]$	$2-\frac{1}{m}+\max\{1, \frac{m-1}{m}\Delta\}$	$3-\frac{1}{m}$
$\mathrm{P}\|prec\|\mathrm{E}\left[\sum w_j C_j\right]$	$(1+\beta)\left(1+\frac{m-1}{m\beta}+\max\{1, \frac{m-1}{m}\Delta\}\right)$	$3+2\sqrt{2}-\frac{1+\sqrt{2}}{m}$
$\mathrm{P}\|r_j, prec\|\mathrm{E}\left[\sum w_j C_j\right]$	$(1+\beta)\left(1+\frac{1}{\beta}+\max\{1, \frac{m-1}{m}\Delta\}\right)$	$3+2\sqrt{2}$

2 Stochastic scheduling

Let us specify the above mentioned dynamic view on stochastic scheduling more precisely; it is based on an embedding of stochastic scheduling into the framework of stochastic dynamic optimization. The *state* of the system at any time t is given by the time t itself as well as the conditional distributions of the jobs' processing times, which depend on the observed *past* up to time t. The past at a time t is given by the set of jobs which have already been completed by t, together with their start and completion times, and the set of jobs which have been started before t but have not been completed yet, together with their start times. The *action* of a scheduling policy at time t is given by a set of jobs $B(t) \subseteq V$ and a tentative decision time $t_{\text{tent}} > t$. The set $B(t)$ is the set of jobs that are scheduled at time t. The tentative decision time t_{tent} is the latest point in time when the next action of the policy takes place, subject to the condition that no other job is released or ends before t_{tent}. Notice that $B(t)$ may be empty, and $t_{\text{tent}} = \infty$ implies that the next action of the policy takes place when the next job is released or some job ends, whatever occurs first. The action of any policy at any time t must only depend on the state of the system at time t. This condition is also called the *non-anticipatory* constraint. Of course, the definition of $B(t)$, with respect to the state at time t, must respect potential release dates and precedence constraints, and the number of available machines. The time instances when a policy takes its actions are called *decision times*. Given an action of a policy at a decision time t, the next decision time is t_{tent}, or the time of the next job completion, or the time when the next job is released, whatever occurs first. Depending on the action of the policy, the state at the next decision time is realized according to the probability distributions of the jobs' processing times.

A given policy eventually yields a feasible m-machine schedule for each realization p of the processing times. For a given policy Π, let $S_j^\Pi(p)$ and $C_j^\Pi(p)$ denote the start and completion times of job j for a given realization p, and let $S_j^\Pi(P)$ and $C_j^\Pi(P)$ denote the associated random variables. It follows from simple examples that, in general, a scheduling policy cannot yield the optimal schedule for each possible realization of the processing times. Hence, our objective is to find a policy Π which minimizes the objective, call it $Z^\Pi(P)$, in expectation. But even under this mild notion of optimality, few special cases exist for which optimal scheduling policies are known. One example is the optimality of the SEPT rule (shortest expected processing time first) for the problem without precedence constraints or release dates, with unit weights, and with exponentially distributed processing times $\mathrm{P}|p_j \sim \exp(\lambda_j)|\mathrm{E}\left[\sum C_j\right]$ (Bruno, Downey, and Frederickson 1981; Weiss and Pinedo 1980). This result was extended by Kämpke (1987) to the case where the weights w_j are compliant with the expected processing times. In general, however, there exist examples which show that optimal policies can be rather complicated in the sense that they must utilize information on the conditional distributions of the jobs' processing times (Uetz 2001, Th. 2.3.8). We therefore concentrate on approximation algorithms. In stochastic scheduling, a scheduling policy Π is said to be an α-*approximation* if its expected performance $\mathrm{E}[Z^\Pi(P)]$ is within a factor of α of the expected performance $\mathrm{E}[Z^{\Pi^*}(P)]$ of an optimal (non-anticipatory) scheduling policy Π^*. The value α is also called the *performance guarantee*.

Let us briefly discuss related models. Compared to the model described above, *on-line optimization* is another way of coping with the fact that the future is uncertain. We refer to Fiat and Woeginger (1998) for details on on-line optimization. There is, however, a significant difference between the underlying paradigms of the above described analysis and the usual competitive analysis that is prevailing in on-line optimization. First, competitive analysis is based upon the ex-post comparison *'What was achieved under uncertainty about the future, and what could have been achieved if the future would not have been uncertain?'*. This is expressed by the fact that the *adversary* of the scheduler is generally an oracle that knows the optimal solution. In contrast, stochastic scheduling addresses the ex-ante question *'What is the best that can be achieved under the given uncertainty about the future?'*. Here, the underlying adversary is much weaker: like the scheduler, the adversary must not anticipate future information. Second, in competitive analysis the adversary is even allowed to determine, to a certain extent, the input distribution. This is not the case in the stochastic model considered here. It is interesting to note that two generalized on-line frameworks were suggested by Koutsoupias and Papadimitriou (2000). They restrict the adversary's power in two ways: Its ability to choose an input distribution, and its ability to find an optimal solution. To a certain extent, the above described stochastic model incorporates both of these underlying ideas. We refer to Koutsoupias and Papadimitriou (2000) for details, and to Uetz (2001) for a discussion. Another type of analysis for stochastic models has been proposed recently by Scharbrodt, Schickinger, and Steger (2002). If $Z^{\mathrm{OPT}}(p)$ is the optimal solution value for a realization p, they analyze the *expected competitive ratio* $\mathrm{E}[Z^\Pi(P)/Z^{\mathrm{OPT}}(P)]$. In this type of analysis the adversary is again an oracle that knows the optimal solution. We refer to Scharbrodt et al. (2002) for a more detailed discussion.

3 List scheduling with deliberate idle times

We start with a few preliminaries and definitions that will be used later in the analysis.

Assumption 3.1. *For any instance of* $P|r_j, prec|*$*, assume that* $r_j \geqslant r_i$ *whenever job* i *is a predecessor of job* j *in the precedence constraints*[†].

Obviously, this assumption can be made without loss of generality. Additionally, we use the following definitions.

Definition 3.2 (critical predecessor). *Let some realization* p *of the processing times and a feasible schedule be given. For any job* j*, a critical predecessor of* j *is a predecessor* i *of* j (*with respect to the precedence constraints*) *with* $C_i > r_j$ *and* C_i *maximal among all predecessors.*

The following definition is illustrated in Figure 1.

Definition 3.3 (critical chain). *Let some realization* p *of the processing times and a feasible schedule be given. For a given job* j*, a critical chain for job* j *and its length* $\ell_j(p)$ *is defined backwards recursively: If* j *has no critical predecessor,* j *is the only job in the critical chain, and* $\ell_j(p) = r_j + p_j$*. Otherwise,* $\ell_j(p) = p_j + \ell_k(p)$*, where job* k *is a critical predecessor of job* j*.*

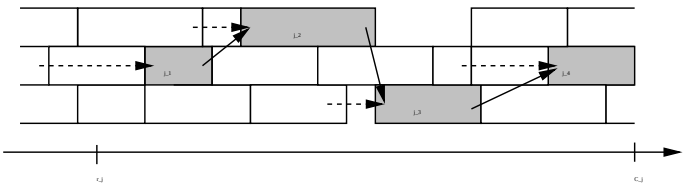

Figure 1: Example of a critical chain for job j. Its length is $\ell_j(p) = r_{j_1} + \sum_{i=1}^{h} p_{j_i}$.

Notice that the critical chain and its length $\ell_j(p)$ depend on both the realization of the processing times p and the underlying schedule. Moreover, since a critical predecessor is not necessarily unique, the critical chain and its length also depend on a tie-breaking rule for choosing critical predecessors. This is not relevant for our analysis, but in order to make the above definition unique, let us suppose that some arbitrary but fixed tie-breaking rule is used. Notice further that, independent of the realization of the processing times, the first job j_1 of a critical chain is available at its release date r_{j_1}. This follows directly from the definition.

To give a precise description of the list scheduling algorithm we use, recall that a job is called available at time t if all predecessors have already been completed by t, and if $t \geqslant r_j$ (whenever release dates are present). Like Graham's list scheduling, the algorithm proceeds over time until all jobs have been scheduled. Assume a priority list L is given. Then, whenever a machine is idle and the first (not yet scheduled) job in the list is available, the job is scheduled. Consider the case that a machine is idle and the first job in the list is not available, but there is some other job available. Then this job is not scheduled immediately as in Graham's algorithm, but it is scheduled only after a certain amount of deliberate idle time has accumulated. Intuitively, the algorithm strives to schedule the jobs in the order of the list L by leaving deliberate idle times, but if the accumulating deliberate idle time exceeds a certain threshold, the algorithm 'panics' and schedules the first available job. To be able to analyze the outcome of the algorithm, the deliberate idle times are charged to the jobs according to a charging scheme. The idea of the charging scheme is that at any time the first available job is 'responsible' for the currently accumulating deliberate idle time, hence this job will be charged the idle time. Algorithm 1 gives a precise description, using the following notation.

Definition 3.4. *For a given realization* p *of the processing times and a given* (*partial*) *schedule, denote by* $r_j(p) \geqslant r_j$ *the earliest point in time when job* j *becomes available.*

Observe that Algorithm 1 coincides with the classical list scheduling algorithm of Graham (1966) if we choose $\beta = 0$, because then the first available job from the list is scheduled at any time.

[†]We use the wildcard $*$ to denote an objective function that may be arbitrary.

Algorithm 1: List scheduling with idle time charging adapted from Chekuri et al. (2001).

input : Instance of $\mathrm{P}|r_j, prec|*$ and a priority list L of the jobs

output : A feasible schedule

initialize $t \leftarrow 0$;
while *there are unscheduled jobs in list* L **do**
  initialize $t_{\text{tent}} \leftarrow \infty$;
  let j be the first unscheduled job in list L;
  if *job* j *is available and a machine is idle at time* t **then**
    schedule job j at t on any of the idle machines;
    charge all uncharged deliberate idle time (from all machines) in $[r_j(p), t]$ to job j;
  else
    let i be the first unscheduled job in list L which is available at time t (if any);
    if *such a job* i *exists and a machine is idle at time* t **then**
      if *there is at least* $\beta \mathrm{E}[P_i]$ *uncharged deliberate idle time in the time interval* $[r_i(p), t]$ **then**
        schedule job i at t on any of the idle machines;
        charge all uncharged deliberate idle time (from all machines) in $[r_i(p), t]$ to job i;
      else
        denote by I be the total amount of uncharged deliberate idle time in $[r_i(p), t]$, let $\overline{m}$ be the number of idle machines at time t, and define t_{tent} such that $I + \overline{m}(t_{\text{tent}} - t) = \beta \mathrm{E}[P_i]$;
        augment t to t_{tent} or the next time when a machine falls idle or a job is released, whatever occurs first;
    else
      augment t to the next time when a machine falls idle or a job is released, whatever occurs first;

Like Chekuri et al. (2001), we prove some properties of the schedules constructed by Algorithm 1. To this end, let us first fix some additional notation. For a given job j, denote by B_j and A_j the sets of jobs that come before and after job j in the priority list L, respectively. By convention, B_j also includes job j. For a given realization p of the processing times and the schedule constructed by Algorithm 1, consider a critical chain $j_1, j_2, \ldots, j_h = j$ for job j. Let us abuse notation and define $B_j(p) := B_j \setminus \cup_{i=1}^{h} \{j_i\}$. For a given realization p and a given schedule, $B_j(p)$ contains all jobs that come before job j in the priority list L, except for those which belong to the given critical chain for j. Moreover, for a given realization p of the processing times, let $O_j(p) \subseteq A_j$ be the jobs in A_j that are started out of order, that is, before j. The following lemma is the analogue to the results for the deterministic setting by Chekuri et al. (2001).

Lemma 3.5. *For any realization p of the processing times and any job j:*

(*i*) *job j is charged no more than $\beta \mathrm{E}[P_j]$ deliberate idle time;*

(*ii*) *the deliberate idle time in $[r_j(p), S_j(p)[$ is charged only to jobs in B_j;*

(*iii*) *there is no uncharged deliberate idle time.*

Proof. Any job j gets charged the deliberate idle time that accumulates (from time $r_j(p)$ on) during time intervals when j is the first available job in the list. In these time intervals, however, j is scheduled as soon as the uncharged deliberate idle time (from time $r_j(p)$ on) accumulates to $\beta \mathrm{E}[P_j]$; this proves (i). In the time interval $[r_j(p), S_j(p)[$ no job from A_j is the first available job from the list, since job j is available from $r_j(p)$ on, and j has higher priority than any job from A_j; this yields (ii). Finally, whenever deliberate idle time accumulates, there is some job which is the first unscheduled and available job in the list L; this proves (iii). □

We next derive an upper bound on the completion time of any job for a given realization p.

Lemma 3.6. *Consider the schedule constructed by Algorithm 1 for any $\beta \geqslant 0$, any realization p of the processing times, and any priority list L which is a linear extension of the precedence constraints. Let $C_j(p)$ denote the*

resulting completion time of any job j, and let $\ell_j(p)$ be the length of a critical chain for job j. Then

$$C_j(p) \;\leqslant\; \frac{m-1}{m}\,\ell_j(p) + \frac{1}{m}r_j + \frac{1}{m}\left(\sum_{i\in B_j}\left(p_i + \beta\,\mathrm{E}[p_i]\right) \;+\; \sum_{i\in O_j(p)} p_i\right). \tag{1}$$

Proof. The basic idea is analogous to Graham's analysis for the makespan objective (Graham 1966). Consider a critical chain for job j with total length $\ell_j(p)$, consisting of jobs $j_1, j_2, \ldots, j_h = j$. Now partition the interval $[r_{j_1}, C_j(p)[$ into time intervals where some job from the critical chain is in process and the remaining time intervals. The latter are exactly $[r_i(p), S_i(p)[$, $i = j_1, \ldots, j_h$. (Recall that $r_{j_1} = r_{j_1}(p)$ due to the definition of a critical chain). By definition,

$$C_j(p) \;=\; \ell_j(p) + \sum_{i=j_1}^{j_h}\left(S_i(p) - r_i(p)\right). \tag{2}$$

To bound the total length of the intervals $[r_i(p), S_i(p)[$, $i = j_1, \ldots, j_h$, observe that in each of these intervals there is no idle time except (possibly) deliberate idle time, since job i is available in $[r_i(p), S_i(p)[$. Hence, the total processing in these intervals can be partitioned into three categories:

- processing of jobs from B_j which do not belong to the critical chain for j, that is, jobs in $B_j(p)$,
- deliberate idle time,
- processing of jobs from A_j which are scheduled 'out of order', that is, jobs in $O_j(p)$.

Due to Lemma 3.5 (ii), all deliberate idle time in the interval $[r_i(p), S_i(p)[$ is charged only to jobs in B_i, $i = j_1, \ldots, j_h$. Since the priority list L is a linear extension of the precedence constraints, we have $B_{j_1} \subset B_{j_2} \subset \cdots \subset B_{j_h} = B_j$. Hence, all deliberate idle time in the intervals $[r_i(p), S_i(p)[$, $i = j_1, \ldots, j_h$, is charged only to jobs in B_j. Since there is no uncharged deliberate idle time (Lemma 3.5 (iii)), and since each job $i \in B_j$ gets charged no more than $\beta\,\mathrm{E}[P_i]$ idle time (Lemma 3.5 (i)), the total amount of deliberate idle time in the intervals $[r_i(p), S_i(p)[$, $i = j_1, \ldots, j_h$, is bounded from above by $\beta\sum_{i\in B_j}\mathrm{E}[P_i]$. This yields

$$\sum_{i=j_1}^{j_h}\left(S_i(p) - r_i(p)\right) \leqslant \frac{1}{m}\left(\sum_{i\in B_j(p)} p_i + \sum_{i\in B_j}\beta\,\mathrm{E}[P_i] + \sum_{i\in O_j(p)} p_i\right). \tag{3}$$

Finally, due to Assumption 3.1 we have $r_{j_1} \leqslant r_j$, thus

$$\sum_{i\in B_j(p)} p_i \;\leqslant\; \sum_{i\in B_j} p_i - \left(\ell_j(p) - r_j\right). \tag{4}$$

Now put (4) into (3), and then (3) into (2), and the claim follows. □

Before we take expectations in (1), we concentrate on the term $\sum_{i\in O_j(p)} p_i$. The following lemma shows that the expected total processing of the jobs in $O_j(p)$ — the jobs that are scheduled out of order with respect to j (and p) — is independent of their actual processing times.

Lemma 3.7. $$\mathrm{E}\left[\sum_{i\in O_j(P)} P_i\right] = \mathrm{E}\left[\sum_{i\in O_j(P)} \mathrm{E}[P_i]\right].$$

Proof. We can write $\sum_{i\in O_j(P)} P_i$ equivalently as $\sum_{i\in A_j}\delta_i(P)\,P_i$, where $\delta_i(P)$ is a binary random variable which is 1 if and only if $i \in O_j(p)$. Linearity of expectation yields

$$\mathrm{E}\left[\sum_{i\in O_j(P)} P_i\right] \;=\; \mathrm{E}\left[\sum_{i\in A_j}\delta_i(P)\,P_i\right] \;=\; \sum_{i\in A_j}\mathrm{E}[\delta_i(P)\,P_i].$$

But $\delta_i(P)$ is stochastically independent of the processing time P_i: when job i is started, it is already decided whether $i \in O_j(p)$, and this decision is independent of the actual processing time of job i. (Here we require that the processing times are stochastically independent, and that policies are non-anticipatory.) Hence,

$$\sum_{i\in A_j}\mathrm{E}[\delta_i(P)\,P_i] \;=\; \sum_{i\in A_j}\mathrm{E}[\delta_i(P)]\,\mathrm{E}[P_i] \;=\; \sum_{i\in A_j}\mathrm{E}\left[\delta_i(P)\,\mathrm{E}[P_i]\right] \;=\; \mathrm{E}\left[\sum_{i\in O_j(P)}\mathrm{E}[P_i]\right]. \tag{5}$$

This concludes the proof of the lemma. □

Finally, we obtain an upper bound on the expected completion time of any job under Algorithm 1.

Lemma 3.8. *For any instance of a stochastic scheduling problem* $P|r_j,prec|*$ *and any priority list L which is a linear extension of the precedence constraints, the expected completion time of any job* j *under Algorithm 1* (*with parameter* $\beta \geqslant 0$) *fulfills*

$$\mathrm{E}[C_j(P)] \;\leqslant\; \Big(\frac{m-1}{m}+\frac{1}{\beta}\Big)\mathrm{E}[\ell_j(P)] \;+\; \frac{1+\beta}{m}\sum_{i\in B_j}\mathrm{E}[P_i] \;+\; \frac{1}{m}r_j\,. \tag{6}$$

Proof. Let j be arbitrary. First, taking expectations in (1) together with Lemma 3.7 yields

$$\mathrm{E}[C_j(P)] \leqslant \frac{m-1}{m}\mathrm{E}[\ell_j(P)]+\frac{1}{m}\left(r_j \;+\; (1+\beta)\sum_{i\in B_j}\mathrm{E}[P_i] \;+\; \mathrm{E}\Big[\sum_{i\in O_j(P)}\mathrm{E}[P_i]\Big]\right).$$

Next, consider any realization p of the processing times. If some job $i\in A_j$ is scheduled out of order, i gets charged exactly $\beta\,\mathrm{E}[P_i]$ idle time. Hence, the total amount of deliberate idle time in $[0,S_j(p)[$ that is charged to jobs in A_j is $\beta\sum_{i\in O_j(p)}\mathrm{E}[P_i]$. Now consider a critical chain for job j, consisting of jobs $j_1,j_2,\ldots,j_h=j$, with total length $\ell_j(p)$. From the proof of Lemma 3.6, we know that all deliberate idle time in the intervals $[r_i(p),S_i(p)[$, $i=j_1,\ldots,j_h$, is charged only to jobs in B_j. In other words, all deliberate idle time in $[0,S_j(p)[$ that is charged to jobs in A_j lies in the complementary intervals $[0,r_{j_1}[$ and $[S_i(p),C_i(p)[$, $i=j_1,\ldots,j_{h-1}$. (Recall that $r_{j_1}=r_{j_1}(p)$ due to the definition of a critical chain.) The total length of these intervals is exactly $\ell_j(p)-p_j$. Hence, the total amount of deliberate idle time in $[0,S_j(p)[$ that is charged to jobs in A_j is at most $m\ell_j(p)$. (In fact, it is at most $m(\ell_j(p)-p_j)$, but this is not essential.) Hence, we obtain $\beta\sum_{i\in O_j(p)}\mathrm{E}[P_i]\leqslant m\ell_j(p)$, for any realization of the processing times. Taking expectations now yields

$$\frac{1}{m}\mathrm{E}\Big[\sum_{i\in O_j(P)}\mathrm{E}[P_i]\Big] \;\leqslant\; \frac{1}{\beta}\mathrm{E}[\ell_j(P)]\,,$$

and (6) follows. □

4 Linear programming relaxation

To obtain a priority list L as input for Algorithm 1, Chekuri et al. (2001) use a single machine relaxation. This approach does not help in the stochastic setting, since the single machine problem does not necessarily provide a lower bound for the parallel machine problem; see Möhring et al. (1999) for an example. Instead, we use LP-relaxations which extend those used by Möhring et al. (1999) by adding inequalities which represent the precedence constraints. First, define $f:2^V\to\mathbb{R}$ by

$$f(W) \;=\; \frac{1}{2m}\Big(\Big(\sum_{j\in W}\mathrm{E}[P_j]\Big)^2 \;+\; \sum_{j\in W}\mathrm{E}[P_j]^2\Big) \;-\; \frac{(m-1)(\Delta-1)}{2m}\Big(\sum_{j\in W}\mathrm{E}[P_j]^2\Big), \qquad W\subseteq V. \tag{7}$$

Here, $\Delta\geqslant 0$ is an upper bound on $\mathrm{Var}[P_j]/\mathrm{E}[P_j]^2$ for all jobs j, where $\mathrm{Var}[P_j]=\mathrm{E}[P_j^2]-\mathrm{E}[P_j]^2$ is the variance of P_j. In other words, the *coefficient of variation*

$$\mathrm{CV}[P_j] := \sqrt{\mathrm{Var}[P_j]}/\mathrm{E}[P_j]$$

of the distribution P_j is bounded by $\sqrt{\Delta}$. The following *load inequalities* are crucial for the derivation of our results.

Theorem 4.1 (Möhring et al. 1999). *If* $\mathrm{CV}[P_j]\leqslant\sqrt{\Delta}$ *for all* P_j *and some* $\Delta\geqslant 0$, *the load inequalities*

$$\sum_{j\in W}\mathrm{E}[P_j]\,\mathrm{E}[C_j^{\Pi}(P)] \geqslant f(W) \tag{8}$$

are valid for all $W\subseteq V$ *and any non-anticipatory scheduling policy* Π.

In fact, an upper bound on the coefficients of variation of the processing time distributions P_j seems to be a reasonable assumption for many scheduling problems. For instance, assume that job processing times follow so-called *NBUE distributions*.

Definition 4.2 (NBUE). *A non-negative random variable X is NBUE, 'new better than used in expectation', if* $\mathrm{E}[X-t|X>t] \leqslant \mathrm{E}[X]$ *for all* $t \geqslant 0$.

Here, $\mathrm{E}[X-t|X>t]$ is the conditional expectation of $X-t$ under the assumption that $X>t$. Roughly spoken, when processing times are NBUE, on average it is not disadvantageous to process a job. Examples for NBUE distributions are, among others, exponential, uniform, and Erlang distributions. A result of Hall and Wellner (1981) states that the coefficient of variation $\mathrm{CV}[X]$ of any NBUE distribution X is bounded by 1. Hence, by choosing $\Delta = 1$ the second term of the right hand side of (8) can be neglected for NBUE distributions, which leads to simplified performance guarantees in Section 5.

Observe that under any scheduling policy Π the trivial inequalities

$$\mathrm{E}[C_j^{\Pi}(P)] \geqslant \mathrm{E}[C_i^{\Pi}(P)] + \mathrm{E}[P_j], \quad (i,j) \in A$$

and

$$\mathrm{E}[C_j^{\Pi}(P)] \geqslant \mathrm{E}[P_j], \quad j \in V$$

are valid, since they even hold point-wise for any realization of the processing times. Due to Theorem 4.1, the following is thus a linear programming relaxation for the problem $\mathrm{P}|r_j, prec|\mathrm{E}[\sum w_j C_j]$.

$$\begin{aligned} \text{minimize} \quad & \sum_{j \in V} w_j C_j^{\mathrm{LP}} \\ \text{subject to} \quad & \sum_{j \in W} \mathrm{E}[P_j] C_j^{\mathrm{LP}} \geqslant f(W), \quad W \subseteq V, \\ & C_j^{\mathrm{LP}} \geqslant C_i^{\mathrm{LP}} + \mathrm{E}[P_j], \qquad (i,j) \in A, \\ & C_j^{\mathrm{LP}} \geqslant \mathrm{E}[P_j], \qquad j \in V, \end{aligned} \tag{9}$$

where $f : 2^V \to \mathbb{R}$ is the set function defined in (7). The load inequalities $\sum_{j \in W} \mathrm{E}[P_j] C_j^{\mathrm{LP}} \geqslant f(W)$, $W \subseteq V$, can be separated in time $\mathrm{O}(n \log n)$ (Möhring et al. 1999; Uetz 2001). Hence, due to the fact that the remaining number of inequalities is polynomial in terms of n, this LP-relaxation can be solved in time polynomial in n by the equivalence of separation and optimization (Grötschel, Lovász, and Schrijver 1988).

The following technical lemma of Möhring et al. (1999) is required later in the analysis.

Lemma 4.3 (Möhring et al. 1999). *Let* $C^{LP} \in \mathbb{R}^n$ *be any point that satisfies the first and the last set of inequalities from* (9). *Assuming* $C_1^{LP} \leqslant C_2^{LP} \leqslant \cdots \leqslant C_n^{LP}$ *we then have*

$$\frac{1}{m} \sum_{k=1}^{j} \mathrm{E}[P_k] \leqslant \left(1 + \max\{1, \frac{m-1}{m}\Delta\}\right) C_j^{LP}$$

for all $j \in V$.

5 Results

We are now ready to prove our results for stochastic machine scheduling problems with precedence constraints.

General precedence constraints. We consider the general problem with precedence constraints and release dates, $\mathrm{P}|r_j, prec|\mathrm{E}[\sum w_j C_j]$. From an optimal solution to the LP-relaxation (9), we define a priority list L according to non-decreasing 'LP completion times' C_j^{LP}. It is perhaps interesting to note that inequalities $C_j^{\mathrm{LP}} \geqslant C_i^{\mathrm{LP}} + \mathrm{E}[P_j]$, $(i,j) \in A$, are only required to ensure that the order according to nondecreasing LP completion times C_j^{LP} is a linear extension of the precedence constraints. They are not required elsewhere in the analysis. Moreover, instead of the weaker inequalities $C_j^{\mathrm{LP}} \geqslant \mathrm{E}[P_j]$ we could as well use $C_j^{\mathrm{LP}} \geqslant r_j + E[P_j]$, but this does not yield an improvement of our results.

Theorem 5.1. *Consider an instance of the scheduling problem* $\mathrm{P}|r_j, prec|\mathrm{E}[\sum w_j C_j]$ *with* $\mathrm{CV}[P_j] \leqslant \sqrt{\Delta}$ *for all* P_j *and some* $\Delta \geqslant 0$. *Let* L *be a priority list according to an optimal solution* C^{LP} *of the linear programming relaxation* (9). *Then Algorithm 1* (*with parameter* $\beta \geqslant 0$) *is an* α*-approximation with*

$$\alpha := (1+\beta)\left(1 + \frac{1}{\beta} + \max\{1, \frac{m-1}{m}\Delta\}\right).$$

Proof. Since L is a linear extension of the precedence constraints, Lemma 3.8 yields

$$\mathrm{E}[C_j(P)] \leqslant \Big(\frac{m-1}{m}+\frac{1}{\beta}\Big)\mathrm{E}[\ell_j(P)] \;+\; \frac{1+\beta}{m}\sum_{i\in B_j}\mathrm{E}[P_i] \;+\; \frac{1}{m}r_j\,,$$

for any job $j \in V$. (Recall that B_j denotes the jobs that come before job j in the priority list L.) Lemma 4.3 yields

$$\frac{1}{m}\sum_{i\in B_j}\mathrm{E}[P_i] \;\leqslant\; \left(1 \;+\; \max\{1, \frac{m-1}{m}\Delta\}\right)C_j^{\mathrm{LP}}\,,$$

for all $j \in V$. Hence,

$$\sum_{j\in V} w_j\,\mathrm{E}[C_j(P)] \leqslant \left(\frac{m-1}{m}+\frac{1}{\beta}\right)\sum_{j\in V} w_j\,\mathrm{E}[\ell_j(P)] + (1+\beta)\left(1 \;+\; \max\{1, \frac{m-1}{m}\Delta\}\right)\sum_{j\in V} w_j C_j^{\mathrm{LP}} + \frac{1}{m}\sum_{j\in V} w_j r_j\,.$$

Now, for any realization p of the processing times, $\ell_j(p) \leqslant C_j(p)$ by definition of a critical chain. Hence, the value $\mathrm{E}[\ell_j(P)]$ is a lower bound on the expected completion time $\mathrm{E}[C_j(P)]$ of any job j, for any scheduling policy. Thus $\sum_{j\in V} w_j\,\mathrm{E}[\ell_j(P)]$ is a lower bound on the expected performance of an optimal scheduling policy. Moreover, both terms $\sum_{j\in V} w_j C_j^{\mathrm{LP}}$ and $\sum_{j\in V} w_j r_j$ are lower bounds on the expected performance of an optimal scheduling policy as well. This gives a performance bound of

$$\left(\frac{m-1}{m}+\frac{1}{\beta}\right) + (1+\beta)\left(1 \;+\; \max\{1, \frac{m-1}{m}\Delta\}\right) + \frac{1}{m}\,.$$

Rearranging the terms yields the desired result. □

Notice that Theorem 5.1 implies a performance bound of $3+2\sqrt{2} \approx 5.828$ if $\beta = 1/\sqrt{2}$ and if the jobs' processing times are distributed according to NBUE distributions (see Definition 4.2).

In fact, the performance bound in Theorem 5.1 can be slightly improved if release dates are absent.

Theorem 5.2. *Consider an instance of the scheduling problem* $\mathrm{P}|prec|\mathrm{E}\left[\sum w_j C_j\right]$ *with* $\mathrm{CV}[P_j] \leqslant \sqrt{\Delta}$ *for all* $j \in V$ *and some* $\Delta \geqslant 0$*. Let L be a priority list according to an optimal solution of the linear programming relaxation* (9)*. Then Algorithm 1* (*with parameter* $\beta \geqslant 0$) *is an* α*-approximation with*

$$\alpha = (1+\beta)\left(1+\frac{m-1}{m\beta}+\max\{1, \frac{m-1}{m}\Delta\}\right)\,.$$

The tighter bound follows from two modifications in the proof of Theorem 5.1. On the one hand, in the proof of Lemma 3.8, one can show that

$$\frac{1}{m}\,\mathrm{E}\left[\sum_{i\in O_j(P)}\mathrm{E}[P_i]\right] \;\leqslant\; \frac{m-1}{m\beta}\,\mathrm{E}[\ell_j(P)]\,.$$

The reason is that there are only $m-1$ machines available for the deliberate idle time that is charged to jobs which are scheduled out of order: Simultaneous to the deliberate idle time, at least a job from the critical chain $j_1, j_2, \ldots, j_h$ is in process. (This argument does not hold if release dates are present, since deliberate idle time could possibly accumulate before r_{j_1}.) On the other hand, it is immediate that the last term $(1/m)\,r_j$ on the right hand side of (6) disappears. With these modifications, the claim follows exactly as in Theorem 5.1.

In-forest precedence constraints. Let us now turn to the special case $\mathrm{P}|$*in-forest*$|\mathrm{E}\left[\sum w_j C_j\right]$. In-forest precedence constraints are characterized by the fact that each job has at most one successor. Moreover, we assume that there are no release dates. For this problem, the results of the preceding section can be further improved.

We start with the following observation which is also contained in (Chekuri et al. 2001, Lemma 10); we give a short proof for the sake of completeness.

Lemma 5.3. *Consider the schedule constructed by Graham's list scheduling for any priority list L which is a linear extension of the* (*in-forest*) *precedence constraints, and any realization p of the processing times. Then, in the interval* $[r_j(p), S_j(p)[$ *there is no processing of jobs in* A_j*.*

Proof. Suppose the claim is false and among all jobs which violate it, let job j be one that is scheduled earliest. Obviously, $S_j(p) > r_j(p)$, otherwise the claim is trivially true. In the interval $[r_j(p), S_j(p)[$ no job from A_j is started, since j is available from time $r_j(p)$ on. Hence, there must be some job $k \in A_j$ that has been started before $r_j(p)$ and that is still in process at $r_j(p)$. Thus $r_j(p) > 0$. Denote by h the number of jobs that are started at time $r_j(p)$. All of these jobs i have higher priority than j, and the fact that j is the first job that violates the claim yields $r_i(p) = r_j(p)$. (At this point it is crucial that the priority list extends the precedence constraints.) In other words, for each of these jobs a critical predecessor ends at time $r_j(p)$, and due to the fact that the precedence constraints form an in-forest, all of these predecessors are different. Hence, including j's critical predecessor, $h+1$ different jobs end at time $r_j(p)$, but only h are started. This is a contradiction since job j is available at time $r_j(p)$. □

Lemma 5.4. *For any* (*stochastic*) *instance of* $P|$*in-forest*$|*$ *and any priority list L which is a linear extension of the precedence constraints, the expected completion time of any job j under Graham's list scheduling fulfills*

$$\mathrm{E}[C_j(P)] \leqslant \frac{m-1}{m}\mathrm{E}[\ell_j(P)] + \frac{1}{m}\sum_{i \in B_j} \mathrm{E}[P_i]\,. \tag{10}$$

Proof. Consider any realization p of the processing times. Given any job j, consider a critical chain for j, consisting of jobs $j_1, j_2, \ldots, j_h = j$ and with total length $\ell_j(p)$. The time interval $[0, C_j(p)]$ can be partitioned into time intervals where a job from a critical chain for j is in process, and the remaining time intervals. Due to Lemma 5.3, in each time interval $[r_i(p), S_i(p)[$ there is no job from A_i in process, for all $i = j_1, \ldots, j_h$. Moreover, there is no idle time on any of the machines in these time intervals (we consider Graham's list scheduling, and there are no release dates). Since $A_{j_1} \supset A_{j_2} \supset \cdots \supset A_{j_h} = A_j$, it follows that the only processing in these time intervals is the jobs in B_j, or more precisely, in $B_j(p)$. In other words, the total processing in these time intervals is at most $\sum_{i \in B_j} p_i - \ell_j(p)$. Hence,

$$C_j(p) \leqslant \frac{m-1}{m}\ell_j(p) + \frac{1}{m}\sum_{i \in B_j} p_i\,,$$

for any realization p. Taking expectations, the claim follows. □

Theorem 5.5. *Consider an instance of* $P|$*in-forest*$|\mathrm{E}[\sum w_j C_j]$ *with* $\mathrm{CV}[P_j] \leqslant \sqrt{\Delta}$ *for all $j \in V$ and some $\Delta \geqslant 0$. Let L be a priority list according to an optimal solution of the linear programming relaxation* (9)*. Then Graham's list scheduling is an α-approximation with*

$$\alpha = 2 - \frac{1}{m} + \max\{1, \frac{m-1}{m}\Delta\}\,.$$

Proof. Graham's list scheduling coincides with Algorithm 1 for $\beta = 0$. The proof is therefore exactly the same as the one of Theorem 5.1, except that Lemma 5.4 is used instead of Lemma 3.8. □

For NBUE distributions (see Definition 4.2), Theorem 5.5 yields a performance guarantee of $3 - 1/m$.

Single machine problems. Theorem 5.2 implies a 2-approximation for the special case of a single machine: In this case the term $(m-1)/(m\beta)$ disappears, and we can choose $\beta = 0$ to obtain performance guarantee 2. (For $\beta = 0$, the algorithm corresponds to Graham's list scheduling.) This holds for arbitrarily distributed, independent processing times. In fact, this matches the best bound currently known in the deterministic setting; see Open Problem 9 in the collection of Schuurman and Woeginger (1999).

6 Further Remarks

A scheduling policy defines a mapping of processing times to start times of jobs. This mapping has to be universally measurable in order to grant existence of the expected objective function value (Möhring et al. 1984). Without going into further details we just mention that the scheduling policies discussed in this paper fulfill this requirement (Uetz 2001, Corollary 3.6.15).

We point out that, apart from the expected processing times of the jobs, a uniform upper bound on their coefficients of variation is the sole stochastic information required as input for the presented scheduling policy. Nevertheless, in our analysis we compare its performance to a lower bound on the performance of *any* non-anticipatory scheduling policy. This refers to the broadest possible sense of scheduling policies as defined by Möhring et al.

(1984). In general, a policy is allowed to take advantage of the *complete* knowledge of the conditional distributions of the processing times, at any time.

Our analysis, however, does not take anticipatory 'scheduling policies' into account since the linear programming lower bound (9) does not hold in this case. This can be seen from the observation that such a 'scheduling policy' could, for instance, compute an optimal schedule for any realization of the processing times, and Theorem 4.1 is no longer true in this case (Uetz 2001). In other words, our analysis is based upon an 'adversary' that is just as powerful as the algorithm itself. This constitutes a major difference compared to the rather 'unfair' competitive analysis known from on-line optimization.

References

Bruno, J. L., P. J. Downey, and G. N. Frederickson (1981). Sequencing tasks with exponential service times to minimize the expected flowtime or makespan. *Journal of the Association for Computing Machinery 28*, 100–113.

Chakrabarti, S. and S. Muthukrishnan (1996). Resource scheduling for parallel database and scientific applications. In *Proceedings of the 8th Annual ACM Symposium on Parallel Algorithms and Architectures*, Padua, Italy, pp. 329–335.

Chandy, K. M. and P. F. Reynolds (1975). Scheduling partially ordered tasks with probabilistic execution times. *Operating Systems Review 9*, 169–177.

Chekuri, C., R. Johnson, R. Motwani, B. Natarajan, B. Rau, and M. Schlansker (1996). An analysis of profile-driven instruction level parallel scheduling with application to super blocks. In *Proceedings of the 29th Annual IEEE/ACM International Symposium on Microarchitecture*, Paris (France), pp. 58–69.

Chekuri, C., R. Motwani, B. Natarajan, and C. Stein (2001). Approximation techniques for average completion time scheduling. *SIAM Journal on Computing 31*, 146–166.

Fiat, A. and G. J. Woeginger (Eds.) (1998). *Online Algorithms: The State of the Art*, Volume 1442 of *Lecture Notes in Computer Science*. Berlin: Springer.

Graham, R. L. (1966). Bounds for certain multiprocessing anomalies. *Bell System Technical Journal 45*, 1563–1581. published in (Graham 1969).

Graham, R. L. (1969). Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics 17*, 416–429.

Graham, R. L., E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics 5*, 287–326.

Grötschel, M., L. Lovász, and A. Schrijver (1988). *Geometric algorithms and combinatorial optimization*, Volume 2 of *Algorithms and Combinatorics*. Berlin: Springer.

Hall, W. J. and J. A. Wellner (1981). Mean residual life. In M. Csörgö, D. A. Dawson, J. N. K. Rao, and A. K. Md. E. Saleh (Eds.), *Proceedings of the International Symposium on Statistics and Related Topics*, Ottawa, Ontario, pp. 169–184. Amsterdam: North-Holland.

Kämpke, T. (1987). On the optimality of static priority policies in stochastic scheduling on parallel machines. *Journal of Applied Probability 24*, 430–448.

Koutsoupias, E. and C. H. Papadimitriou (2000). Beyond competitive analysis. *SIAM Journal on Computing 30*, 300–317.

Möhring, R. H., F. J. Radermacher, and G. Weiss (1984). Stochastic scheduling problems I: General strategies. *ZOR - Zeitschrift für Operations Research 28*, 193–260.

Möhring, R. H., A. S. Schulz, and M. Uetz (1999). Approximation in stochastic scheduling: The power of LP-based priority policies. *Journal of the Association for Computing Machinery 46*, 924–942.

Munier, A., M. Queyranne, and A. S. Schulz (1998). Approximation bounds for a general class of precedence constrained parallel machine scheduling problems. In R. E. Bixby, E. A. Boyd, and R. Z. Ríos-Mercado (Eds.), *Proceedings of the 6th International Conference on Integer Programming and Combinatorial Optimization*, Houston (TX), Volume 1412 of *Lecture Notes in Computer Science*, pp. 367–382. Berlin: Springer. Journal version: SIAM Journal on Computing, to appear.

Scharbrodt, M., T. Schickinger, and A. Steger (2002). A new average case analysis for completion time scheduling. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, Montréal (Québec), pp. 170–178.

Schulz, A. S. (1996). *Polytopes and Scheduling*. Ph. D. thesis, Institut für Mathematik, Technische Universität Berlin, Berlin, Germany.

Schuurman, P. and G. J. Woeginger (1999). Polynomial time approximation algorithms for machine scheduling: Ten open problems. *Journal of Scheduling 2*, 203–213.

Uetz, M. (2001). *Algorithms for Deterministic and Stochastic Scheduling*. Ph. D. thesis, Institut für Mathematik, Technische Universität Berlin, Berlin, Germany. Published in: Cuvillier Verlag, Göttingen, Germany.

Weiss, G. and M. Pinedo (1980). Scheduling tasks with exponential service times on non-identical processors to minimize various cost functions. *Journal of Applied Probability 17*, 187–202.